ISSN 2233-5382 © 2022 KODISA
JIDB website: http://www.jidb.or.kr
doi:http://dx.doi.org/10.13106/jidb.2022.vol13.no12.59

Heuristics for Carrying-out of Export Containers in Container Terminal

Young-Kyu PARK[1]

Received: September 21, 2022. Revised: November 05, 2022. Accepted: December 15, 2022.

Abstract

Purpose: Re-handling is an important factor that reduces the productivity of container terminals. The purpose of this paper is to propose an algorithm to find the order of container movement in order to minimize the number of re-handling in the process of carrying-out. **Research design, data and methodology**: This paper proposes an algorithm to set the optimal carry-out order and conducted tests to evaluate the performance of the algorithm proposed in this paper. 1. tests comparing the performance of an algorithm proposed in an existing study with that proposed in this paper. 2. Performance tests for bays with complex structures. **Results**: Test 1 shows that the algorithm proposed in this paper performs better than the existing algorithm. Test 2 shows that the proposed algorithm can also be used in bays with considerably high complexity and that there is no major problem with using it in the field. **Conclusion**: While we can conclude that the proposed algorithm as a carry-out algorithm is more effective than conventional methods, research is needed on how to handle more complex bays more effectively. This is because the larger the bay, the more container combinations increase, making it difficult to find the best carry-out order.

Key words: Carry-out, Rehandling, Container, Terminal Yard

JEL Classifications: C61, L91, C88, R49.

1. Introduction

한 나라의 수출입과 관련하여 중요한 역할을 수행하는 물류거점으로 컨테이너 터미널이 있다. 컨테이너 터미널은 육상운송과 해상운송의 연결점 역할을 하는데, 수출의 경우 컨테이너가 트럭을 통하여 컨테이너 터미널로 들어오는 육상운송과, 선박으로 적재된 후 컨테이너 터미널 밖으로 나가는 해상운송이 컨테이너 터미널에서 이루어지고, 수입의 경우 그 반대 방향으로 해상운송과 육상운송이 이루어진다. 전 세계적으로 컨테이너 물동량의 증가와 그에 따르는 컨테이너 터미널의 개발로, 국가 간과 터미널 간의 경쟁이 치열해 지고 있다. 그래서 컨테이너 터미널은 이런 경쟁에서 우위에 서기 위해서 터미널의 생산성을 높이도록 노력해야 한다. 수출의 경우, 컨테이너가 컨테이너 터미널에 들어오면 장치장에 일단 보관되는데, 장치장과 관련한 작업은 크게 3가지로 나눌 수 있다. 컨테이너를 장치장에 넣는 반입 작업과 반입된 컨테이너를 정돈하는 재정돈 작업 그리고 컨테이너를 장치장에서 꺼내는 반출 작업이 그것이다. 컨테이너를 장치장에 넣을 때 효율적으로 공간을 활용하기 위하여 컨테이너 여러

1 First and Corresponding Author. Professor, Department of Business & Logistics, Kaya University, Korea,
Email: ykpark@kaya.ac.kr

개를 하나의 단으로 쌓게 되는데 이로 인하여 재취급(re-handling)이라는 비생산적인 작업이 발생하게 된다. 수출컨테이너의 경우 반입 작업시 반입 순서는 컨테이너가 장치장에 도착하는 순서에 달려있게 된다. 반출 작업의 반출 순서는 컨테이너의 무게를 기준으로 하는 선박의 적재 순서에 따른다. 이런 반입 순서와 반출 순서의 차이 때문에, 위에 언급한 재취급이 발생하게 된다. 재취급이 발생하면 선반의 재항시간이 늘어나서 결국 터미널의 생산성을 저해하게 된다. 본 연구는 수출컨테이너를 대상으로 장치장에서 반출 작업을 수행할 때 효과적인 반출 순서를 찾는 알고리즘을 대상으로 한다. 수출 컨테이너가 반입이 완료되고 선박의 적재 순서가 정해진 후 그리고 반출을 시작하기 전에 허용되는 시간내에 최소의 재취급 횟수로 반출하기 위한 컨테이너 이동 순서를 찾는 것이 목적이다. 논문의 구성은 2장에서 관련 연구에 대하여 알아보고, 본 연구에서 제안하는 알고리즘의 특성에 대하여 언급한다. 3장에서 연구방법론에 대하여 기술하며 본 논문이 제안하는 알고리즘에 대하여 설명한다. 4장에서는 제안하는 알고리즘의 성능 측정을 위한 실험을 실시하고, 5장에서 연구에 대한 요약과 시사점을 언급하며, 논문의 결론을 맺고자 한다.

2. Literature Review

2.1. 관련 연구

컨테이너 장지장의 성능에 영향을 미치는 재취급과 관련한 여러가지 연구가 있는데, 먼저 공컨테이너 재취급과 관련한 연구를 알아본다. Lee and Kim(2019)은 공 컨테이너를 적하할 때 발생할 수 있는 재취급의 횟수에 대한 연구를 수행하였다. 공 컨테이너 장치장을 운영하는 방법과 사양 그리고 사용 장비 등이 공 컨테이너를 적하할 때 발생할 수 있는 재취급 기대횟수에 미치는 영향에 대한 연구로, 여러 선사의 여러 유형의 공 컨테이너가 혼재한 상황에서 발생할 수 있는 재취급의 기대값을 계산하는 다양한 수식을 유도하였다. Lee and Kim(2018)은 공 컨테이너를 반출할 때 발생하는 재취급의 횟수에 대한 연구를 수행하였는데, Lee and Kim(2019)와 마찬가지로 공 컨테이너에 대한 재취급 횟수에 대한 연구이다. 앞의 연구와의 차이점은 앞의 연구는 적하 작업을 대상으로 실시하였고, 이 연구는 반출 작업을 대상으로 하였다는 점이다.

수출 컨테이너와 관련한 세가지 장치장 작업 중 가장 먼저 수행되는 반입작업이 장치장의 성능에 큰 영향을 미친다고 볼 수 있다. 반입작업의 결과에 따라 장치장의 컨테이너들의 배치가 결정되고, 그 배치 결과에 따라 반출작업시 재취급 발생횟수가 거의 결정되기 때문이다. 반입작업이 끝난 후 선박이 도착하기 전까지 재정돈 작업을 수행해서 반출시의 재취급 발생을 완전히 없앨 수 있으므로 재정돈 작업의 수행여부도 성능에 큰 영향을 미친다. 재정돈 작업이 수행되지 않았을 경우 영향을 미치는 것은 마지막 작업인 반출 작업이 된다. 결국 재취급 발생은 반출 작업을 수행하는 동안 발생하기 때문이다.

반입과 관련하여 장치장에 미치는 영향에 관한 연구로 Kim and Bae(2001)은 선박의 입출항과 외부트럭의 반출입을 분석하여, 수출컨테이너를 대상으로 장치장의 소요 공간 산정에 대한 연구를 수행하였다. Park and Kwak (2011)은 반입 시에 사전 작업을 수행함으로써 반출 시점의 재취급을 줄이는 휴리스틱을 제안하였는데, 이 방안의 효과를 시뮬레이션을 통해서 증명하였다. Kang et al. (2004)은 반입시점에 컨테이너 무게 정보를 안다는 가정하에, 반입시 컨테이너 적재 위치를 결정함으로써 반출 시점의 재취급을 줄이는 효과적인 휴리스틱 알고리즘을 제시하였다. 본 논문에서도 성능 실험을 하는데 필요한 반입 과정에 이 알고리즘을 적용하여 실험을 하였

다.

다양한 연구들이 있었으나 반입 작업으로 재취급을 줄이는 데는 한계가 있다. 반입시점에는 해당 컨테이너의 선박으로의 적재 순서가 정해지지 않고, 반입순서도 마음대로 정할 수 없기 때문이다.

다음으로 재정돈과 관련하여 상당히 많은 연구가 이루어 졌다. 재정돈 작업은 반출작업에서 재취급이 발생하지 않도록 적재를 다시 하는 작업으로 반입이 완료된 후에 실시하게 된다. Bae et al. (2006)은 수출 컨테이너에 대하여 반출 작업을 수행하기 전에 장치장에 흩어져 있는 컨테이너들을 같은 공간에 재배치하는 작업인 이적 작업에 대한 연구를 수행하였다. 이 연구에서 컨테이너 터미널에서 수행되는 이적 작업 비용을 줄이고, 선박에 적재하는 작업의 생산성을 높이는 방안을 제시하였다. 그리고 Oh et al. (2005)도 여러 블록들에 흩어져 있는 컨테이너들을 한 곳에 모으는 방안을 연구하였는데 교차가 불가능한 복수 트랜스퍼 크레인의 활용 방법에 대하여 촛점을 맞추었다. 이 연구에서 복수 트랜스퍼 크레인들 간의 간섭을 최소화하면서 작업 시간을 줄일 수 있는 방안을 제시하였다. Ha and Kim (2012)은 재정돈을 하나의 베이 내에서 수행하는 방안에 대한 연구를 하였는데, 컨테이너 재배치 문제를 도입해서, A* 알고리즘 (Nilsson, 1998)을 기본으로 한 최적 이적 순서를 수립하는 방법을 개발하였다. 베이의 복잡도가 높을 경우 재정돈 해를 찾지 못하는 경우도 있었지만, 일반적인 복잡도의 베이에 대해서는 재정돈 해를 찾을 수 있다는 것을 증명하였다. Park (2016)은 이웃한 베이의 상단에 있는 빈 슬롯을 활용하는 재정돈 방안에 대하여 연구를 수행하였으며, 복잡도가 상당히 높은 베이에 대해서도 재정돈의 해를 찾을 수 있었다. 시뮬레이션을 통하여 성능을 실험하였다. Park et al. (2010)는 재정돈 작업이 상당한 시간이 소요되므로 충분한 시간을 확보하지 못했을 때 선택적으로 재정돈을 실시하는 방안에 대하여 연구를 수행하였다. 연구 방법으로 휴리스틱 방안과 유전알고리즘에 기반을 둔 방안들을 비교하였고, 각 방안들에 대한 장단점을 분석하였다.

재정돈에 대한 연구가 다양한 조건에서 상당히 많이 이루어진 것은 재정돈이 재취급을 수행할 경우 재취급이 전혀 발생하지 않도록 배치를 완전히 다시 할 수 있어서 장치장의 성능에 미치는 영향이 크기 때문일 것이다. 그러나 재정돈의 경우에는 상당히 많은 작업이 필요하고 많은 시간이 필요하다. 그리고 베이의 크기가 클 경우 한 베이 내에서 수행하는 재정돈의 경우 재정돈 순서를 파악하기가 쉽지 않다.

재정돈을 수행하지 못할 경우 재취급을 해결해야 것은 반출 작업이 된다. 반출 작업에서는 컨테이너들의 배치에 따라 재취급을 전혀 발생하지 않도록 하는 것은 불가능하며 가능하면 최소로 발생하도록 할 수 밖에 없다. 반출과 관련한 연구로 Lee and Lee (2010)는 여러 베이에 흩어져 있는 컨테이너들을 대상으로 레일 장착 갠트리 크레인으로 반출을 수행하는 알고리즘에 대하여 연구를 수행하였는데, 컨테이너 이동 시간과 갠트리 크레인의 이동시간의 합을 줄이기 위한 알고리즘을 제안하였다. 이 논문에서는 이진 정수계획법과 혼합 정수계획법을 사용하였다. Kim. and Hong (2006)는 단일 베이의 컨테이너를 대상으로 효과적으로 반출하는 방안에 대하여 연구를 하였다. 반출을 실행하는 도중에 재취급이 발생할 때, 그 시점에 해당 컨테이너를 이동할 위치를 찾는 방안으로 Branch & Bound 알고리즘과 간단하고 효과적인 휴리스틱을 제안하였다. Branch & Bound 알고리즘의 경우는 위치를 찾는 시간이 너무 오래 걸려 현장에서 적용하기 어려움이 있고, 제안한 휴리스틱은 위치를 찾는 시간은 적게 걸리나 Branch & Bound 알고리즘 보다 효과가 적은 결과가 나왔다. Wan et al. (2009)도 단일 베이의 컨테이너를 대상으로 효과적으로 반출하는 방안에 대한 연구를

수행하였는데, 정수계획법을 기반으로 하는 휴리스틱을 제안하였다. 반입이 완료된 베이에 대하여 반출을 수행할 때 재취급시 컨테이너를 장치할 위치를 결정할 때 효과적임을 시뮬레이션으로 보였다. 지금까지의 관련 연구들은 대부분 재취급이 발생했을 때 해당 컨테이너를 배치할 위치를 찾는 방법을 제시하였으며, 알고리즘을 수행하는데 상당히 많은 시간이 소요되는 경우가 많이 있었다. 그리고 베이의 복잡도가 높지 않아 현장에서 활용하기에는 문제들이 있었다. 본 연구에서는 현장에 적용할 수 있는 반출 휴리스틱을 제안한다. 그래서 이 휴리스틱으로 짧은 시간내에 복잡도가 높은 베이에 대하여 효과적인 반출 순서를 찾을 수 있음을 보인다. 또한 재취급 예측 알고리즘을 제시하여 반출 과정 중에 있는 베이에서 반출이 완료될 때까지 발생하게 될 재취급을 예측하는데 사용한다.

2.2. 관련 용어

컨테이너 터미널에 컨테이너가 들어오면 장치장에 보관하게 되는데 효율적인 공간활용을 위하여 컨테이너를 여러 단으로 쌓아서 보관하게 된다. 하나의 컨테이너 위에 여러 단을 모아 놓은 것을 스택이라고 하며, 스택을 여러 개 모아 놓은 것을 베이라고 한다. 여러 베이들을 묶어서 블록이라고 부르는데, 아래 그림 Figure 1은 하나의 블록 중 일부를 나타내고 있는데, 블록 중 2개의 회색 베이와 하나의 흰색 베이만 묘사하고 있다. 스택을 '열'로, 단을 '행'으로 표현하기도 하는데 이 그림은 4단 5열의 베이를 나타내고 있으며, 제일 앞의 베이에 있는 5개의 스택 중 빗금 친 부분에 3단으로 된 스택의 예를 보이고 있다.

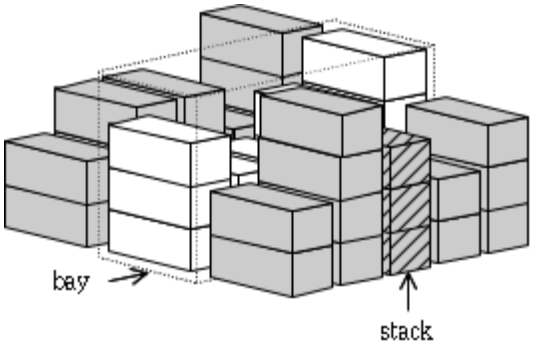

Figure 1: A Part of a Block

우선 순위란 컨테이너들을 반출하는 순서로써 인덱스라고도 하며 선박에 적재하기 위하여 컨테이너를 장치장에서 꺼내는 순서에 해당한다. 보통 컨테이너의 무게를 기준으로 정해지는데 컨테이너의 무게가 무거울수록 높은 우선순위를 부여한다. 이는 먼저 무거운 컨테이너를 선박의 아래쪽에 장치함으로써 선박의 안정성을 높이기 위한 것이다.

3. Research Methods

먼저 이 논문의 반출 알고리즘들을 설명하기 위하여 사용하게 될 용어와 관련 기호들을 언급한다.

C_N : 다음에 반출하려는 컨테이너
P_N : 컨테이너 C_N의 우선순위
P_i^k : i열 k행에 있는 컨테이너의 우선순위
mP_i^k : i열에서 0행에서 k행 사이에 있는 컨테이너들 중 가장 큰 우선순위
H_i : i열에 적재된 컨테이너 개수
H : 한 베이 내의 열의 높이, 적재할 수 있는 컨테이너 최대 개수
W : 한 베이 내의 열의 개수
$L_{i,j}$: i열에서 j열까지의 컨테이너 이동거리, 즉 |j-i| 이며, i와 j가 인접한 열일 경우 1이 된다.
$Cn(\ )$: ( )의 조건을 만족하는 컨테이너의 수

3.1. 재취급 문제

본 연구의 대상인 수출 컨테이너의 재취급은 반

출과정에서 발생하게 된다. 베이는 H단까지 쌓여 있는 W개의 스택으로 구성되어 있고, 우선순위로 표현된 반출 순서에 따라 반출 작업을 수행한다. 하나의 베이에서 컨테이너들을 반출하는 과정을 보면, 먼저 반출할 컨테이너('목표 컨테이너')를 찾게 되는데, 목표 컨테이너의 상단에 다른 컨테이너가 놓여 있는가 놓여있지 않는가에 따라서 재취급이 발생하는가 하지 않는가가 결정된다. 목표 컨테이너 위에 다른 컨테이너가 놓여있지 않으면, 재취급없이 목표 컨테이너를 정상 반출하고, 목표 컨테이너 위에 다른 컨테이너들이 놓여 있으면, 어쩔 수 없이 위에 놓여 있는 다른 컨테이너들을 다른 스택으로 이동하는 재취급 작업들을 먼저 한 후 목표 컨테이너를 반출해야 한다. 반출 과정에서 재취급이 발생하게 되는 과정을 Figure 2에 W가 3, H가 4인 베이로 나타내고 있다. 컨테이너는 숫자로 표현된 사각형으로 나타내었고, 빈 사각형은 빈 슬롯을 나타낸다. 숫자의 값은 우선순위를 의미하고 낮은 값이 먼저 반출된다.

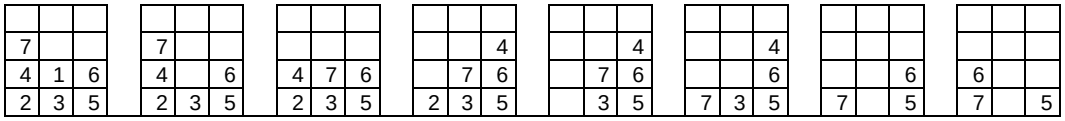

Figure 2: Example 1 of Carry-out Sequence

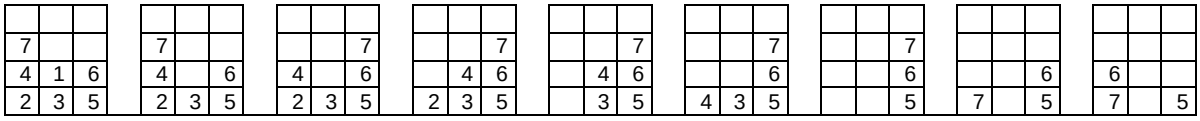

Figure 3: Example 2 of Carry-out Sequence

먼저 베이 (a)에서 목표 컨테이너인 우선순위 1인 컨테이너(2번째 스택, 2번째 단에 있는)를 반출해야 하는데 상단에 다른 컨테이너가 없으므로 재취급없이 반출을 수행하게 되고, 반출 결과로 베이 (b)가 된다. 베이 (b)에서 우선순위2인 컨테이너를 반출할 때 상단에 우선순위 7과 4인 컨테이너들이 있으므로 이들을 다른 곳으로 먼저 이동을 해야한다. 먼저 7을 2번째 스택으로 옮겨 베이 (c)가 되는데 이때의 컨테이너 이동이 재취급이다. 베이 (c)에서 우선순위 4인 컨테이너를 3번째 스택으로 이동시켜서 베이 (d)가 된다. 이런 과정으로 진행을 해나는 것이 가면 재취급이 모두 4회 발생한 후(재취급이 발생하는 이동 : (b)->(c), (c)->(d), (e)->(f), (g)->(h)), 베이 (h)가 되는데, 베이 (h)에서는 남아 있는 컨테이너들이 순서대로 반출될 때 상단에 컨테이너가 없으므로 더 이상 재취급없이 반출을 완료할 수 있게 된다.

반출 과정에서 고려할 점은, 재취급이 발생하는 이동은 어쩔 수 없다하더라도, 추가 재취급으로 이어지는 이동은 피해야 한다는 것이다. Figure 2와 같은 베이지만 반출 과정을 Figure 3와 같이 실행하면 재취급 횟수는 5회가 된다(재취급이 발생하는 이동 : (b)->(c), (c)->(d), (e)->(f), (g)->(h), (h)->(i)). 컨테이너 7의 경우, Figure 2와 Figure 3에서 모두 추가로 재취급이 발생하므로, 컨테이너 7로 인한 재취급 발생 회수는 모두 2가 된다. 그러나 컨테이너 4의 경우에는 Figure 2에서는 재취급이 발생한 후 추가로 재취급이 발생하지 않아 재취급 횟수가 1이지만 Figure 3에서는 추가로 재취급이 발생하여 재취급 횟수는 2가된다. 그래서 두 Figure에서 전체 재취급 횟수에 차이가 발생하게 된다. 추가로 재취급이 발생하지 않도록 스택을 선택하여 이동할 수 있으면 그 열로

이동을 하는 것이 중요한 이유가 여기에 있다.

추가로 재취급이 일어나느냐 일어나지 않느냐는 것은 재취급 발생으로 컨테이너를 이동할 때, 어느 스택(열)로 이동하느냐에 달려있다. 다음은 추가로 재취급이 일어나지 않는 열을 선택하는 기준이다. 열j에 있는 컨테이너 C_N을 재취급으로 인하여 다른 열로 이동을 해야 할 경우, 한번의 재취급으로 끝나는 이동 대상이 되는 열i는 다음 조건 (1)과 (2)를 만족하는 열이다.

$$H_i < H, \text{ 단 } j \neq i \quad (1)$$

$$mP_i^H < P_N, \text{ 단 } j \neq i \quad (2)$$

$$min_{i=1}^{W}\{mP_i^H\}, \text{ 단 } j \neq i \quad (3)$$

위의 조건 (1)은 상단에 이동 가능한 공간이 있는 열을 의미하며, 조건(2)는 그 열의 최대 우선순위가 이동할 컨테이너의 우선순위보다 낮은(숫자는 더 큰) 열을 의미한다. 조건(3)은 조건(2)를 만족하는 열들 중 최대 우선순위를 갖는 열을 의미한다. 만약 위의 조건(1)과 조건(2)를 만족하는 열이 하나 이상일 경우에는 위의 조건(3)을 만족하는 열로 이동하는 것이 향후 반출과정에서 발생할 추가 재취급 횟수를 최소화 할 가능성이 크다. 이는 Kang et al. (2004)이 제안한 반입알고리즘에서, 반입시 적재하는 열을 선택할 때 재취급을 효과적으로 줄이는 열을 선택하는 방안인데, 본 논문에서 반출과정에서 재취급 작업을 수행할 때 이동할 열을 선택하는 방안으로 도입을 하였으며, 이 방안으로 열을 선택하면 향후 반출과정에서 재취급 발생하는 것을 효과적으로 줄일 수 있다고 본다.

만약 위의 조건(1)과 조건(2)를 만족하는 열이 없다면 추후 재취급이 발생할 수 밖에 없는 열로 컨테이너를 이동할 수 밖에 없다. 위에서 언급한 것처럼 한번의 재취급 발생은 막을 수는 없지만 추가로 재취급이 발생되는 횟수는 최소가 되도록 하는 것이 중요하다. 다음은 본 논문에서 제안하는 알고리즘에 대하여 알아본다.

3.2. ABS(Astar Based Search) 알고리즘

이 알고리즘은 본 논문에서 제안하는 알고리즘으로, 반출과정에서 목표 컨테이너가 스택의 가장 상단에 있어서 바로 반출할 수 있으면, 바로 반출을 하지만, 목표 컨테이너 위에 다른 컨테이너들이 놓여 있으면 컨테이너 이동에 대한 모든 경우에 대하여 검토를 해서, 재취급 횟수에 대한 정보를 바탕으로 전체적으로 가장 재취급 횟수가 적게 발생하는 이동을 선택하는 방식을 사용한다. 이를 위하여 A* 알고리즘을 이용하는데, 베이의 컨테이너 적재 상태를 하나의 노드로 나타내어, 컨테이너의 이동으로 인한 노드의 변화 과정을 평가하여 최종적으로 재취급이 최소가 되는 노드의 변화, 즉 경로를 찾는 방식을 사용한다. 이 알고리즘을 사용하여 재취급 횟수가 가장 적은 경로의 노드를 선택해 가면 결국 재취급 횟수가 가장 적은 방법으로 반출을 완료할 수 있게 된다. 여기서 재취급 횟수에 대한 정보는 평가값으로 나타내며 자세한 설명은 아래에서 한다. 먼저 본 논문에서 제안하는 ABS 알고리즘의 구체적인 흐름에 대하여 설명한다.

3.2.1. 알고리즘의 흐름

먼저 이 알고리즘을 위하여 베이의 컨테이너 적재 상태를 하나의 노드로 나타내고, 알고리즘을 시작하면 초기 베이 상태를 첫 노드로 설정하여 아래 알고리즘 구조로 진행을 하게 된다.(이 절차는 A*알고리즘(Nilsson, 1998; Ha & Kim, 2012)을 기본으로 하여, 본 알고리즘에 맞게 변경하였음)

[알고리즘 구조]

1. 빈 OPEN리스트와 빈 CLOSED리스트를 준비한 후 OPEN리스트에 첫 노드를 넣는다.
2. OPEN리스트의 제일 앞 노드n 을 꺼낸다. 만약 OPEN 리스트가 비어 있으면 알고리즘이 실패로 종료된다.

3. 노드n에서 재취급없이 반출이 가능한 컨테이너가 있는 동안 계속 반출한다.
4. 노드n의 모든 컨테이너가 반출이 되었으면 알고리즘이 성공으로 종료한다.
5. 노드n을 CLOSED리스트에 넣고, 노드n의 한 컨테이너를 다른 열로 이동하는 작업으로 자식 노드의 집합M을 만든다. 집합M에 속한 노드m에 대하여 다음 작업들을 수행한다.
   1) 노드m에서 재취급없이 반출이 가능한 컨테이너가 있는 동안 계속 반출한다.
   2) 노드m의 모든 컨테이너가 반출이 되었으면 알고리즘이 성공으로 종료한다.
   3) 노드m과 동일한 노드m'가 CLOSED리스트 또는 OPEN리스트에 있는지에 따라 아래 해당 작업을 수행하고 6단계로 간다.
   - CLOSED리스트에 있으면, 노드m을 파기한다.
   - OPEN리스트에 있고, 노드m의 평가값이 노드m'의 평가값보다 크거나 같으면, 노드m을 파기한다.
   - OPEN리스트에 있고, 노드m의 평가값이 노드m'의 평가값보다 작으면, 노드m을 노드m'로 대체한다.
   - CLOSED리스트와 OPEN리스트에 모두 없으면, 노드m을 OPEN 리스트에 삽입한다.
6. OPEN리스트에 있는 노드들은 평가값으로 정렬한다. 평가값을 정하는 기준은 아래 와 같다.
   1) 평가값 ($f(n)$의 값) 을 기준으로 오름차순
   2) 동일한 평가값일 경우, 재취급 횟수($g(n)$의 값)을 기준으로 내림차순
7. 단계2로 가서 작업을 계속한다.

이 알고리즘은 OPEN리스트와 CLOSED리스트를 준비하면서 시작하는데, 여기서 OPEN리스트는 검색 대상이 되는 노드를 저장하기 위한 리스트이고, CLOSED리스트는 검색이 완료된 노드를 저장하기 위한 리스트이다. 먼저 1단계에서 첫 노드를 OPEN 리스트에 넣는다. 첫 노드를 대상으로 재취급없이 목표 컨테이너의 반출이 가능한지 확인해 보고, 가능하다면 가능한 동안 목표 컨테이너를 계속 반출을 해나간다. 만약 진행해 나가다가, 재취급 없는 반출이 안 될 경우에 자식 노드를 만드는 과정을 거치게 된다. 자식 노드의 생성 작업은 노드의 한 열 위에 있는 컨테이너를 다른 열로 옮기는 작업으로써 하나의 자식 노드를 생성하게 되는데, 모든 열 위의 컨테이너에 대하여 이동 가능한 모든 다른 열로 이동함으로써 자식 노드의 그룹을 생성하게 된다. 그리고 노드에는 컨테이너 이동할 때 마다 그 내용을 기록해 둠으로써 전체 컨테이너 이동 단계를 알게 된다. 이렇게 생성된 그룹내의 각 자식 노드에 대하여 위의 5단계 3)에 있는 4가지 작업 중 하나를 수행하게 되는데, 첫 줄은 노드m과 같은 노드가 CLOSED리스트에 있는 경우, '이미 동일한 노드로 5번까지의 과정을 거친 적이 있고 그래서 두 번 진행할 필요가 없다'는 의미이고, 두번째 줄은 '이전에 동일한 노드m'로 이미 평가를 한 적이 있고, 그 평가값이 노드m의 평가값보다 좋으므로, 노드m'를 노드m으로 대체하여 진행할 필요가 없다'는 뜻이다. 여기서 '동일한 노드'라는 것은 베이내의 컨테이너 배치가 같다는 의미인데, 동일한 노드들이 평가값이 다를 수 있는 이유는 컨테이너 이동은 다른 순서로 그리고 다른 횟수로 진행되어도, 결과적으로 컨테이너 배치가 같아질 수 있고 그로 인하여 이동 횟수를 의미하는 평가값은 달라질 수 있기 때문이다. 세번째 줄은 '이전에 동일한 노드m'로 이미 평가를 한 적이 있고, 그 평가값이 노드m의 평가값보다 나쁘므로, 노드m'를 노드m으로 대체하여 진행할 필요가 있다'는 뜻이다. 네번째 줄은 '두 리스트에 없는 노드이므로 새로운 노드이고 앞으로 평가를 진행해야한다'의 의미이다. 다음 단계인 6단계로 넘어가면 OPEN리스트 정렬 작업을 수행하게 되는데, 이 정렬이 평가값이 좋은 노드부터 작업을 하도록 만드는 것이니까 결국 재취급이 적은 노드가 먼저 평가가 되고, 결국 재취급이 적은 노드로 알고리즘을 종료하게 된다. 그리고는 2단계로 돌아

가서 다음에 작업할 노드를 선택하면서 위의 과정을 반복하게 된다. 이 알고리즘은 컨테이너를 반출하면서 수행되므로, 알고리즘이 수행될수록 노드에 있는 컨테이너 개수가 점점 줄어 들는 상황이 발생하게 된다. 그래서 알고리즘이 진행되면서 모든 컨테이너가 다 반출되는 노드가 나타나면 그 노드가 가장 컨테이너 이동이 적은, 즉 재취급이 가장 적게 반출하는 노드가 된다. 이 때가 알고리즘의 정상 종료 상황이 되며, 모든 노드들을 다 처리하여도 모든 컨테이너가 반출되는 노드가 없을 때 알고리즘은 실패하게 된다. 알고리즘이 실행되는 과정을 보이기 위하여 Figure 4에, 앞에서 살펴 본 Figure2의 (a) 베이를 첫 노드로 시작해서 진행해 가는 과정의 일부를 나타내었다

Figure 4에서 각 노드는 표와 표 상하에 글자들 표시되어 있고, 각 표의 상단에는 좌측에 자신의 번호, 우측에는 부모 노드를 표시하였다. 예를 들면 3번째 줄 첫 노드의 1, fr:0은 자신이 1번 노드이고 부모 노드는 0 이라는 의미이다. 노드에서 자식노드로의 연결은 화살표로 표시하였다. 위의 3번째 노드는 부모 노드로 0노드와 연결되어 있고 자식 노드로 5, 6과 연결되어 있다. 표의 하단에는 평가값을 표현하였다. 평가값은 3.2.2에서 자세히 소개한다. 노드를 선택하는 기준이 평가값이 되기 때문에 같은 이동 횟수를 가지는 같은 가로줄 노드들 중에서 평가값이 적은 노드가 먼저 검색이 된다. 예를 들면 5번에서 8번노드들 중에서 6번과 8번 노드의 평가값이 4(f = g+h = 2+2 = 4)이고, 5번과 7번노드의 5보다 적어서 먼저 방문이 된 것을 볼 수 있다. 회색으로 표현된 노드들은 Figure 2에서 보았던 반출 과정에 해당하는 노드들을 나타낸다.

위에서 언급한 '동일한 노드'와 관련하여 정규 형태에 하여 언급하고자 한다. 알고리즘을 수행하면서 자식노드들을 많이 생성하게 되는데, 그로 인하여 두 리스트에 많은 노드가 연결되므로 많은 저장공간이 필요해진다. 각 노드의 컨테이너 배치를 그대로 사용하지않고 정규 형태로 만들어서 사용함으로써 두 리스트의 저장공간을 줄이고 알고리즘의 수행 시간 낭비를 줄일 수 있다.

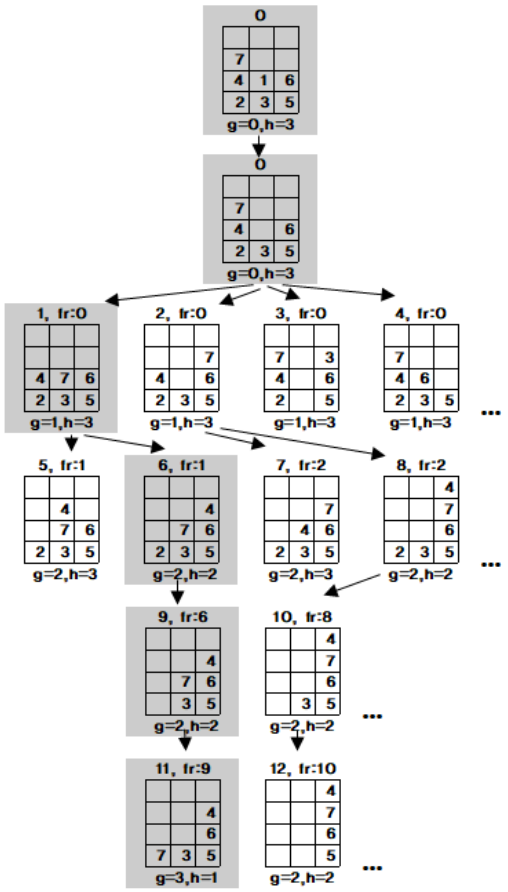

Figure 4: Process of Executing ABS Algorithm

정규 형태는 스택들을 오름차순으로 정렬한 형태로 나타낸다. 즉, 각 스택을 아래 단에서 시작해서 위의 단으로 올라가면서, 장치된 컨테이너의 인덱스를 사용한 수열로 간주하고, 스택들을 그 수열의 오름차순으로 정렬한다. 빈 슬롯은 인덱스 0의 컨테이너가 있다고 가정한다. 예를 들어 아래 Figure 5 베이 (a)의 장치 형태는 1-2-0-0, 1-0-0-0, 4-2-3-0로 생각할 수 있다. 이 장치 형태를 정규 형태를 만들기 위하여 스택들을 오름차순으로 정렬하면 베이 (e)와 같이 1-0-0-0, 1-2-0-0, 4-2-3-0이 될 것이다. 즉,

베이 (e)가 베이 (a)의 정규 형태가 된다. 컨테이너의 배치가 다르더라도 정규 형태가 같으면 반출 알고리즘실행 결과도 같다. 아래 Figure 5의 베이 (a),(b),(c),(d) 모두 정규 형태는 베이 (e)가 되므로 모두 다 같은 노드가 되며, 이와 같이 정규 형태를 사용하면, 컨테이너 배치는 달라도 실제로는 같은 노드들의 중복 저장과 중복 수행을 줄일 수 있다. 이 정규 형태는 Kim et al(2000)과 Ha and Kim (2012)에서 사용된 방법이다.

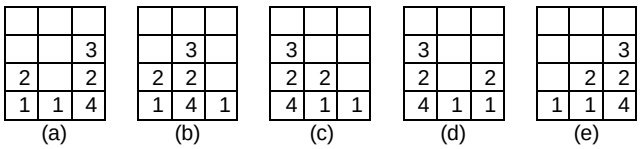

Figure 5: Normalization for Layouts

3.2.2. 평가 함수

위의 알고리즘에서 언급한 평가값은 아래 (4)에 보인 것처럼 $g(n)$과 $h(n)$의 합인 평가 함수 $f(n)$으로 구하는데, $g(n)$은 첫 노드에서 노드n까지 실시한 전체 재취급의 횟수를 의미하고, $h(n)$은 노드n에서 마지막 노드(모든 컨테이너가 반출된)까지 수행하리라 예상되는 전체 재취급의 횟수를 의미한다.

$$f(n) = g(n) + h(n) \qquad (4)$$

즉 한 노드n의 평가값은 첫 노드에서, 노드n을 거쳐, 모든 컨테이너를 반출할 때까지 발생하게 되는 전체 재취급 횟수를 나타낸다. 반출 과정에서 스택의 제일 상단에 있는 컨테이너는 이동없이 반출하게 되므로 평가값에는 영향을 미치지 않는다. 제일 상단에 있지 않은 컨테이너를 반출할 때 컨테이너 이동이 발생하게 되므로 평가값에 영향을 미치게 된다.

평가값을 구하는 방법을 보면 먼저 $g(n)$은 첫 노드에서 시작해서 노드n 이 될 때까지 수행한 컨테이너 이동의 수를 세면 되므로 구하기는 간단하다. 그리나 $h(n)$은 노드n 에서 시작해서 반출이 완전히 끝날 때까지 수행하게 될 컨테이너 이동 횟수를 예측해야 하므로 구하기가 쉽지가 않다. $h(n)$의 값은 두 수의 합으로 구한다. 첫번째 수는 노드n의 상태에서 1회라도 재취급을 해야 하는 컨테이너 수이며, 두번째 수는 그 컨테이너가 재취급을 수행할 때 추가된 재취급 횟수이다. 먼저 첫번째 수를 구하는 방법에 대하여 이야기 한다. 재취급을 해야 하는 컨테이너는 아래 식(5)을 만족하는 컨테이너들인데, 자신보다 아래에 있는 컨테이너들 중 자신보다 우선순위가 높은 컨테이너가 하나라도 있으면 재취급을 해야만 한다.

$$P_i^k < mP_i^{k-1} \qquad (5)$$

이는 반출 과정이 우선순위를 기준으로 반출해 나가므로, 자기 아래에 있는 자기보다 높은 우선순위의 컨테이너가 자신보다 먼저 반출되어야 하기 때문이다. 한 노드에서 재취급 컨테이너의 개수는 식(6)로 구할 수 있다. 식(6)은 식(5)을 만족하는 컨테이너의 수를 모든 열에 대하여 구해서 더한다는 의미이며, 한 노드의 컨테이너 적재 상태에서 바로 계산으로 구할 수 있다.

$$\sum_{i=1}^{W}\sum_{k=1}^{H} Cn(P_i^k < mP_i^{k-1}) \qquad (6)$$

다음으로 두번째 수를 구하는 방법에 대하여 이야기 한다. 위의 식(5)에 해당되는 컨테이너를 재취급할 때 위에서 언급한 것처럼 위의 식(1)과 식(2)를 만족하는 스택이 없어서, 식(1)과 식(2)를 만족하지 못하는 스택으로 이동할 때, 그 컨테이너의 아래에 놓여있는 컨테이너의 상황에 따라서 나중의 어느 시점엔가 추가로 재취급을 해야만 한다. 한 노드에서 추가로 발생하게 되는 재취급의 횟수는 첫번째 수처럼 식으로 간단히 구하기가 어렵다. 왜냐하면, 추가로 발생하는 재취급은 컨테이너가 반출되어가는 과정 속에서 나중 어느 시점엔가 발생하게 되는데, 컨테이너가 반출되어 가면서 변할 수 있는 베이

의 구조는 너무 다양해져서 언제, 그리고 몇 번 재취급이 일어나게 될지 파악하기가 너무 어렵기 때문이다. 그래서 이 논문에서는 휴리스틱을 이용하여 노드 n의 추가 재취급 횟수를 포함하여 전체 재취급 횟수를 구하는데, 아래 그 휴리스틱에 대한 소개를 한다.

3.2.3. 재취급 예측 휴리스틱

이 알고리즘은 ABS알고리즘을 수행하면서 노드의 평가값 중 $h(n)$을 구할 때 사용하는 휴리스틱이다. 이 알고리즘으로, 한 노드에서 시작해서 모든 컨테이너를 반출할 때까지 발생하게 될 재취급 횟수와 추가 재취급 횟수, 즉 전체 재취급 횟수를 예측할 수 있다. 대상이 되는 노드에서 시작해서 종료시까지 반출을 직접 수행하면서 발생하는 최소 재취급 횟수를 측정하는 방식으로 구한다.

[재취급 예측 휴리스틱]

1. 빈 임시 풀을 준비한 후, 노드 n 의 복사본 노드 n’ 를 만든다.
2. 노드 n’ 의 다음 목표 컨테이너가 스택의 가장 상단에 있거나, 임시 풀에 있는 동안 계속 반출한다.
3. 노드 n’ 의 모든 컨테이너가 반출이 되었으면 재취급 수를 반환하고 알고리즘을 종료한다.
4. 노드 n’ 의 목표 컨테이너 위에 다른 컨테이너들이 있으면, 가장 상단의 컨테이너부터 목표 컨테이너 바로 위의 컨테이너까지 차례대로 선택하여 다음 작업들을 수행한다.
   1) 스택의 최고 우선순위가 선택한 컨테이너의 우선순위보다 같거나, 낮은 스택들이 있고 이동 가능한 공간이 있다면, 그 중 가장 우선순위가 높은 스택으로 선택한 컨테이너를 이동한다.
   2) 스택의 최고 우선순위가 선택한 컨테이너의 우선순위보다 같거나, 낮은 스택이 없다면, 선택한 컨테이너를 별도의 임시 풀로 이동하고 재취급의 횟수를 1증가한다.
5. 2단계로 간다.

단계4에서 2)의 경우에 선택한 컨테이너를 별도의 풀로 이동시키는데 이렇게 풀로 이동된 컨테이너는 그곳에 있다가 자신의 반출 순서가 되면 반출하게 된다. 이렇게 하는 이유는 별도의 풀이 아닌 다른 열로 이동시키는 것이 상당히 복잡한 상황을 만들기 때문이다. 즉, 어떤 열로 이동하는가에 따라 추가 재취급의 발생하는 횟수가 변할 수 있으므로 정확한 재취급 횟수를 구하기 힘들고, 추후 상황의 변화가 너무 심하여 정확한 재취급 횟수를 구하기 어렵기 때문이다. 이 방식으로 반출을 하면 정확한 재취급 횟수는 구하지 못하지만 최소 재취급 발생 횟수는 구할 수 있게 된다.

4. Experiment and Result

4.1. 실험 환경과 대상

이 논문에서 제안하는 알고리즘의 성능을 보이기 위하여 여러가지 구성의 베이에 대하여 반출알고리즘의 성능을 측정한다. 알고리즘의 성능을 측정하기 위하여 각 알고리즘을 C++로 구현하였으며 구현하기 위한 프로그래밍 툴과, 실험을 수행한 컴퓨터 시스템 사양은 아래와 같다.

프로그래밍 툴 : Microsoft Visual C++ 2022
운영체제 : Windows 11 Pro, 프로세서 : Intel(R) Core(TM) i3-8100 CPU 3.60 GHz, RAM : 16.GB

그리고 알고리즘의 성능을 측정하기 위하여 여러 가지 베이들을 대상으로 실험을 수행하였다.

4.2. 기존 연구 결과와 비교

Table 1 은 의미가 있다고 생각되는 연구와 비교한 결과를 보여준다. 기존의 연구로 본 논문과 같은 주제를 다룬 연구 중 Wan et al. (2009)의 연구를 대상으로 비교를 실시하였다. Table 1에서 사용한 용어로 W는 스택의 수, H는 단의 수, C는 컨테이너 수를

의미한다. Average Number은 알고리즘을 수행했을 때 발생한 평균 재취급 수로 전체 재취급 수를 베이수로 나눈 값이다. CPU Time은 알고리즘을 수행했을 때 소요된 평균 시간으로 단위는 초이며 다음에 나오는 Table들에서도 같은 의미이다. 기존의 연구에서는 스택과 단의 하나의 조합에 대하여 컨테이너 수를 3가지로 나누어서 실험을 하였는데, Light, Medium, Heavy로 각각의 개수는 $\lceil 0.2((W-1)H+1) \rceil$, $\lceil 0.5((W-1)H+1) \rceil$, 그리고 $\lceil 0.8((W-1)H+1), \rceil$로 구했다. 여기서 $\lceil a \rceil$는 a보다 작지 않은 가장 작은 정수를 의미한다. 이 연구에서는 단의 높이가 2인 베이들((W, H, C)가 (6, 2, 3), (6, 2, 6), (6, 2, 9))과 Light에 해당하는 베이들((W, H, C)가 (6, 3, 4), (6, 4, 5), (6, 5, 6))을 실험에서 제외하였는데 이 베이들의 복잡도가 너무 낮아서 큰 의미가 없다고 생각했기 때문이다.

Table 1: Performance Comparison between Two Algorithms

W	H	C	Previous Results		Our Results	
			Average Number	CPU Time	Average Number	CPU Time
6	3	8	1.76	0.077	1.32	0.013
6	3	13	4.58	0.432	3.91	0.012
6	4	11	3.36	0.327	2.70	0.013
6	4	17	7.56	85.974	7.21	0.013
6	5	13	4.72	4.538	3.97	0.014
6	5	21	11.68	689.421	11.16	0.015

실험대상이 되는 베이는 실험결과를 보면 모든 베이에 대하여 이전 연구에 비하여 제안한 알고리즘이 평균 재취급횟수가 낮은 것을 볼 수 있다. 그리고 실행시간 역시 짧다는 것을 알 수 있다.

4.3. 복잡도를 높인 베이 실험

다음은 시험 대상 베이의 복잡도를 더 높여서 본 논문의 알고리즘 성능을 검정하였다. 스택(W)은 8,9,10으로 단은 각 스택에 따라서 4,5,6,7,8,9까지 사용하였다. 컨테이너의 수는 4.2에서 실시한 조건과 같이 Medium과 Heavy에 해당하는 베이들로 무작위로 100개씩 생성하여 실험을 실시하였다.

Table 2: Experimental Results on More Complex Bays

W	H	Medium			Heavy		
		C	Average Number	CPU Time	C	Average Number	CPU Time
8	4	15	3.70	0.015	24	10.54	0.016
8	5	18	5.98	0.017	29	14.90	0.018
8	6	22	8.59	0.019	35	21.90	0.030
8	7	25	11.76	0.025	40	27.25	0.035
9	4	17	4.13	0.014	27	11.43	0.017
9	5	21	6.88	0.017	33	16.79	0.022
9	6	25	9.81	0.020	40	24.15	0.031
9	7	29	13.25	0.027	46	30.79	0.048
9	8	33	16.77	0.039	52	38.99	2.290
10	5	23	7.47	0.018	37	19.50	0.026
10	6	28	11.53	0.024	44	25.94	0.047
10	7	32	14.40	0.031	52	35.40	0.134
10	8	37	19.15	0.045	59	43.34	0.295
10	9	41	22.98	0.064	66	53.27	7.431

실험결과를 보면 베이의 스택과 단 그리고 컨테이너 수가 4.2의 실험에 비하여 상당히 커진 것을 볼 수 있다. 그럼에도 수행시간은 크게 증가하지는 않았다는 것을 알 수 있다. 스택과 단의 조합이 (9,8)과 (10,9) 인 때를 제외하면 1초 미만이라는 것을 알 수 있다.

4.4. 반입 알고리즘을 거친 베이에 대한 실험

위에서는 기존의 연구에서 제안한 조건으로 실험을 수행하였으나, 여기서는 효과적인 반입 알고리즘을 거친 베이에 대하여 실험을 하여 성능을 측정해 보고자 한다. 수출 건테이너의 경우 효과적인 반입 알고리즘으로 반입 작업을 수행할 경우 재취급 발생을 효과적으로 줄일 수 있었으며(Kang et al. 2004), 이 알고리즘으로 반입을 한 베이에 대하여 제안하는 알고리즘으로 반출할 경우의 성능을 검증하고자 한다. 대상 베이들은 열과 단을 기준으로 8열 6단, 9열 7단, 10열 8단의 3종류 베이들이며, 이들 베이들에 대하여 다시 컨테이너의 수와 인덱스 수를 아래 Table 3과 같이 변화하면서 실험 대상 베이

들을 선정하였다. 복잡도가 높은 베이를 대상으로 실험하기 위하여, 컨테이너 수는 각 종류별로 그 베이에서 적재가 허용되는 최대수에서 하나의 단의 수만큼 뺀 수로 구했으며 식으로는 ((W − 1) H)가 되며 복잡도가 거의 최대에 해당한다. 앞에서 실시한 실험에서 Heavy의 베이를 $\lceil 0.8((W-1)H+1) \rceil$로 구한 것에 비하면 복잡도가 상당히 높다는 것을 알 수 있다. 최대 복잡도에 해당하는 값을 기준으로 하나씩 줄여 나가며 4개씩 생성하였다. 그리고 인덱스는 컨테이너 수와 동일하게 부여하는 것으로 선정하였으며 이 역시 최대의 복잡도를 만들기 위해서였다.

Table 3: Bay Specification for Experiment

W	H	No. of Container
8	6	39, 40, 41, 42
9	7	53, 54, 55, 56
10	8	69, 70, 71, 72

이렇게 선정된 베이들에 대하여 100개씩 무작위로 생성하여 제안한 알고리즘의 성능을 측정하였다. 실험 대상이 되는 베이는 반입과정에서 컨테이너가 장치장으로 반입될 때는 임의의 순으로 반입되지만 Kang et al. (2004)이 제안한 반입알고리즘으로 반입하였으며 이는 현실적인 반입알고리즘을 가정하였기 때문이다.

Table 4: Experimental Results after Executing Carry-in Algorithm

W	H	C	I	Average Number	CPU Time
8	6	39	39	4.06	<0.1
		40	40	5.38	<0.1
		41	41	5.37	<0.1
		42	42	6.34	<0.1
9	7	53	53	7.85	<0.1
		54	54	9.04	<0.1
		55	55	9.97	<0.1
		56	56	11.30	<0.1
10	8	69	69	14.12	<0.1
		70	70	15.31	<0.1
		71	71	15.41	<0.1
		72	72	18.26	<0.1

Table 4에 사용한 약자는 앞의 Table들과 같고, I의 경우 우선순위의 개수를 의미한다. I가 C와 같다는 것은 모든 컨테이너들이 다른 우선순위를 갖는다는 것을 의미하며, 또한 컨테이너들의 반출순서가 서로 다르다는 것을 뜻하며 이 역시 복잡도를 높이기 위해서이다. 실험 결과를 전체적으로 보면 단과 스택의 수가 증가하면 재취급 수가 점점 커지는 것을 알 수가 있고, 같은 단과 스택일 경우, 컨테이너 수가 증가하면 역시 재취급 수가 점점 커지는 것을 알 수가 있다. 이는 복잡도가 커질수록 재취급 횟수가 증가한다는 것이고 이는 당연한 결과라고 볼 수 있다. 스택이 10이고 단이 8인 경우에도 알고리즘을 실행하는데 소요된 시간은 베이 하나에 대한 평균 시간을 의미하는데 대부분 0.1초를 넘기지 않는다는 것을 알 수 있다. 이는 반출하기 전에 알고리즘을 수행하는 점을 감안하지 않더라도 현장에서 적용하는데 문제가 없음을 의미한다.

5. Conclusions and Discussion

컨테이너 터미널의 성능에 영향을 미치는 수출 컨테이너에 대한 작업으로 반입과 재정돈, 그리고 반출 작업이 있다. 그리고 이들 작업에서 재취급 발생을 줄이는 것이 컨테이너 터미널의 성능에 아주 중요한 요소이다. 본 논문에서는 재취급 발생을 줄이기 위한 반출 알고리즘들을 다루었다. 반출 과정의 알고리즘으로 A*를 기반으로 하는 ABS알고리즘을 개발하여 효과적인 반출 방법을 제안하였다. 그리고 ABS알고리즘에 재취급 횟수를 구하는 문제를 해결하는 휴리스틱을 도입하였다. 이 알고리즘의 평가를 위하여 기존의 연구와 비교하는 실험을 수행하였고, 제안한 알고리즘이 실행시간은 더 작고 재취급의 횟수 역시 더 작은 것을 알 수 있었다. 그리고 제시한 알고리즘의 전반적인 성능을 평가하기 위하여 기존의 연구와 같은 조건으로 더 복잡도가

높은 베이들과 그리고 최고의 복잡도를 갖는 반입 알고리즘을 실시한 베이들에 대하여 성능을 평가하였다. 그 결과 제안한 알고리즘이 현장에서 사용하는데 큰 문제가 없다는 결론을 내릴 수 있었다. 본 논문에서 제안하는 알고리즘에 대한 아쉬운 점은 제안한 알고리즘이 최적의 방식은 아니라는 점이다. 베이 내의 컨테이너의 수가 많아질수록 발생할 수 있는 반출 순서의 조합은 기하급수적으로 커지기 때문에, 그 많은 조합에 대하여 최적의 반출 순서, 즉 최소 재취급 횟수로 반출을 수행하는 순서를 찾아내는 것은 어려움이 있다. 그리고 그 많은 조합에 대하여 실행하는 시간을 줄이는 것도 큰 과제이다. 이 두가지 문제는 서로 연관이 있으며, 이를 해결하기 위해서는 평가값 중 재배치 값을 더욱 짧은 시간에 더욱 정확하게 구하는 방법과, A*알고리즘을 대체할 알고리즘이나 능가하는 검색 기법을 도입하는 방법이 있을 것으로 생각된다. 그래서 반출 반출 알고리즘에 가장 적합한 검색 기법을 찾는 연구가 필요하다고 생각한다. 그리고 재정돈 알고리즘에 대한 기존 연구에서 이웃한 베이의 빈 슬롯을 활용하는 방법에 대한 연구가 있었는데, 향후 연구에서는 반출 알고리즘에도 이웃한 베이의 빈 슬롯을 활용하여 성능을 높이는 방안에 대한 추가적인 연구를 해보는 것도 이 문제를 다른 방향으로 해결한다는 의미에서 의미가 있으리라 생각된다.

References

Bae, J. W., Park, Y. M. & Kim, K. H., (2006). Export Container Remarshaling Planning in Automated Container Terminals, *The Korean Operations Research and Management Science Society, 2006 Spring Conference*, 1186 – 1193.

Ha, B. H., & Kim, S. S. (2012). A* Algorithm for Optimal Intra-bay Container Pre-marshalling Plan, *Journal of the Korean Institute of Industrial Engineers*, *38*(2), 157-172.

Kang, J. H., Oh, M. S., Ru, K. R., & Kim, K. H. (2004), Method of Inbound Container Positioning for Minimal Rehandling Considering Weight, *Proceedings of The 2004 KIISS Fall Conference*, 271-278.

Kim, K. Y. & Bae, J. W., (2001). A Model for Determining the Space Requirement for Export Containers, *Journal of Dongseo University Research Center*, *4*, 19-30.

Kim, K. H., Park, Y. M., and Ryu, K.-R. (2000). Deriving Decision Rules to Locate Export Containers in Container Yards, *European Journal of Operational Research*, *124*, 89-101.

Kim, K. H. & Hong, G. P., (2006). A heuristic rule for relocating blocks, *Computers & Operations Research*, *33* (2006) 940-954.

Lee, Y. S. & Lee, Y. J., (2010). A heuristic for retrieving containers from a yard, *Computers & Operations Research*, *37* (2010) 1139-1147.

Lee, H. & Kim, K. H., (2018). Comparing Expected Numbers of Re-Handles for Empty Containers During Gate-Out Operation, *Journal of Korean Navigation and Port Research*, *42*(3), 207-216.

Lee, H. & Kim, K. H., (2019). Estimating the Expected Number of Re-handles for Empty Containers during Loading Operation, *Journal of Korean Navigation and Port Research*, *43*(3), 197-208.

Nilsson, N. J. (1998). *Artificial Intelligence : A New Synthesis*, San Francisco, California, USA, Morgan Kaufmann Publishers, Inc., pp. 142-160.

Oh, M. S., Kwang, J. H., Yu, K. R., & Kim, K. H. (2005). A Heuristic Approach to Scheduling Multiple Cranes for Intra-Block Remarshalling, *Journal of Korean Navigation and Port Research, 29*(5), 447-455.

Park, Y. K. (2016), Remarshalling Plan Using Neighboring Bay in Container Terminal, *Journal of Korean Navigation and Port Research, 40*(3), 113-120.

Park, Y. K., & Kwak, K. S. (2011). Export container preprocessing method to decrease the number of rehandling in container terminal, *Journal of Korean Navigation and Port Research*, *35*(1), 77-82.

Park, K. Y., Park, T. J. & Ryu, K. R. (2010). Iterative Container Reselection Methods for Remarshaling in a Container Terminal, *Journal of Korean Navigation and Port Research*, *34*(6), 503-509.

Wan, Y. w., Liu. J. & Tsai, P. C. (2009). The assignment of storage locations to containers for a container stack, *Naval research logistics*, *56*(8), 699-713.